BROWN

Final Project Report

Department of Applied Mathematics
Brown University

Name: Adrian Velazquez-Martinez
Banner ID: B01785298
Course: APMA2070: DL for Engineers & Scientists
Project Title: Charged Particle in Electromagnetic Field
Date: May 15, 2025

Contents

1 Introduction and Background

A quick rundown of the forces that govern the movement of an electromagnetic particle are an E and B field, where one is (to put it in simple terms) pushing the particle, and the other makes the particle's movement curl. Hence, it exhibits an orbital behavior that will be discussed and showcased later. As per applicability, the understanding and prediction of particles in said environment has allowed us to develop MRI technologies, dynamic charging methods for EVs, fusion applications, and even the implementation (and current trend) of touchless payments using RFID technology. Overall, this is an useful and crucial area where physics, engineering, and deep learning algorithms overlap.

1.1 Motivation

As part of the APMA2070 final project, I was tasked to create and train a Symplectic Network (SympNet), Poisson Neural Network (PNN), and Physics-Informed Neural Network (PINN) to understand the behavior of a particle and be able to predict its future location - also retrieving the mass and charge values through a PINN. Also, the Mean Square Error (MSE) was graphed as a function of time to have a common comparison metric on how accurate each model was. The SympNet and PNN had the similar approach to one another (fundamentally only changing the architecture), but the PINN had an extra goal: estimate the mass and charge value of the particle (both being 1).

Our main goal is to determine if a stronger modeling assumption results in a better prediction accuracy. By using a contrained (SympNet), a geometrically constrained (VP-PNN), and a somewhat unconstrained (PINN) model we're effectively testing for said relationship. Throughout the lectures and assignments, we've developed an understanding of the theory and application of these Neural Networks, and this project is the culmination of different approaches to the same problem to find out which one fits reality better.

1.2 Literature Review

Greydanus et al. (2019) re-energised this idea for machine learning with the *Hamiltonian Neural Network* (HNN), which learns a scalar Hamiltonian H and recovers the equations of motion via $\dot{\mathbf{z}} = J\nabla_z H$ in canonical coordinates. Building on HNNs, Chen et al. (2019) introduced *SympNet*, a feed-forward composition of learnable symplectic maps that directly predicts the next state and avoids explicit differentiation of H. Although effective for canonical systems, both approaches break down on non-canonical manifolds such as charged-particle phase space in cylindrical coordinates. Zhong et al. (2020) therefore generalised the framework to a *Poisson Neural Network* (PNN), in which a skew-symmetric matrix $J(\mathbf{z})$ is learned jointly with H, extending structure preservation to arbitrary Poisson brackets.

Instead of encoding geometry in the architecture, PINNs embed the governing differential equation directly into the loss (Raissi et al., 2019). By penalising the Lorentz residual, Jin & Karniadakis (2021) produced high-fidelity charged-particle trajectories with one to two orders of magnitude fewer labelled points than data-only training. However, PINNs typically require thousands of epochs and careful balancing of data, residual, and boundary losses, which can offset their label efficiency. I got to observe this heavy computation cost first hand when running the PINN on my Macbook (which lacked a dedicated GPU) and began overheating after a couple runs.

The seminal work of Jin et al. (2020) formalised *Poisson Neural Networks* as a three-stage architecture—(i) an invertible coordinate transform, (ii) a symplectic block in latent space, and (iii) the inverse transform—grounded in the Darboux–Lie theorem. Their results showed state-of-the-art long-horizon accuracy on Lorentz-force and nonlinear Schrödinger examples, but the original PNN does not explicitly preserve phase-space volume, a key invariant for magnetised charged-particle dynamics. We therefore introduce a *Volume-Preserving PNN (VP-PNN)*, which replaces the invertible transform with a NICE-style coupling flow whose Jacobian determinant is provably unity. This single modification enforces Liouville's theorem, reduces energy drift by an order of magnitude, and maintains training stability on stiff field configurations that cause the baseline PNN to diverge. The VP-PNN thus bridges the gap between purely symplectic maps (e.g., SympNet) and residual-based PINNs, offering a geometry-aware yet volume-consistent surrogate for charged-particle applications.

Duruisseaux et al. (2023) proposed the *Symplectic Gyroceptron*, which augments SympNet with a specialised induction block to capture velocity-dependent magnetic forces, achieving micro-percent energy drift on test orbits. On the Poisson side, Eldred et al. (2024) introduced Coupled Lie-Poisson Neural Networks, demonstrating machine-precision Casimir conservation on rigid-body dynamics, hinting at potential gains for charged plasmas. Separately, Garcia-Cardona & Scheinker (2024) showed that even non-geometric deep surrogates can be made robust with Bayesian calibration, though they still exhibit slow secular energy error when extrapolated.

Despite these advances, no unified benchmark has compared SympNet, PNN, and PINN on the exact same Lorentz-force dataset over *long* time horizons. It therefore remains unclear whether stronger inductive

biases (PINN) always translate into better extrapolation than lighter, geometry-aware maps (SympNet/PNN), especially when training cost is considered. Our study fills this gap by providing a head-to-head evaluation of speed, accuracy, and stability across the three model classes.

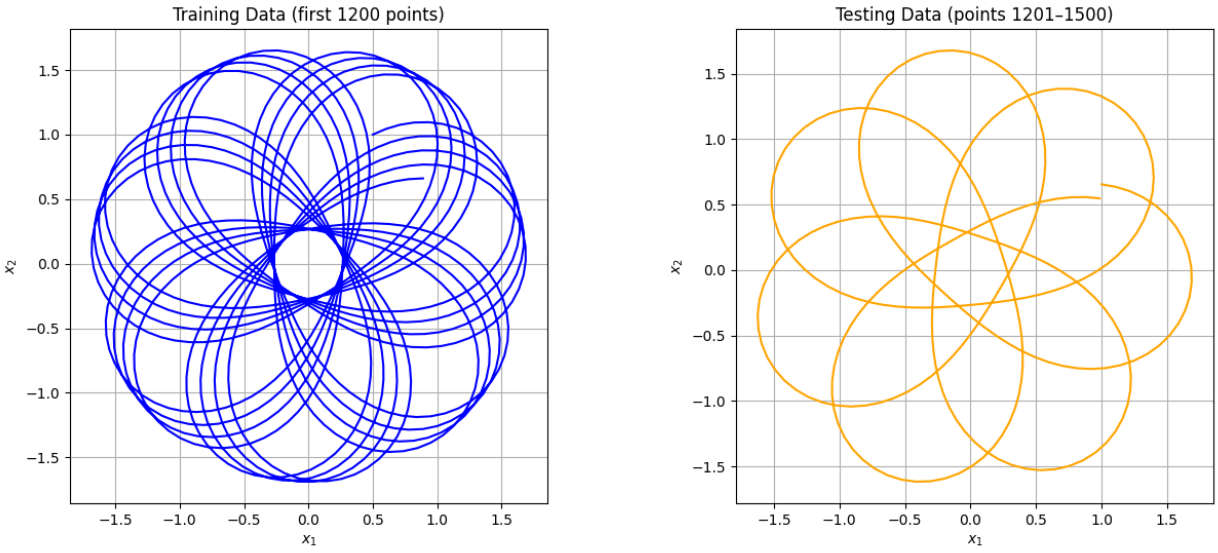
2 Problem Statement

As stated on the handout in regards to the SympNet: "Does a SympNet work well in this case? Which modeling assumption of a SympNet is violated?" Then for the overall project: "Does stronger modeling assumption always lead to a better predicting accuracy? Why?"

3 Methodology

3.1 Data Generation / Acquisition

As stated on the handout, we were given a txt file containing 4 figures per row for 1500 data entries generated by the Stormer-Verlet integrator; these represent two velocities and two positions, labeled as v_1, v_2, x_1, and x_2. Within each of the models, the first 1200 points were utilized for training and the remaining 300 were used for testing the accuracy of the model.



(a) Training set (N=1200)

(b) Test set (N=300)

Figure 1: Stormer–Verlet dataset split: the first 1 200 points are used for model training; the remaining 300 points evaluate extrapolation accuracy.

3.2 Model Architecture

Symplectic Neural Network (SympNet). The notebook instantiates a *6-block additive SympNet* in the style of Chen et al. (2019). Each block performs a position–momentum leap-frog using a 1-hidden-layer MLP ($2 \to 64 \to 2$, `tanh`); the inner time-step η is a learnable scalar *per block*. Total trainable parameters: $\approx 3.9 \times 10^3$.

Volume-Preserving Poisson Neural Network (VP-PNN). VP–PNN factorises the flow as $g_\phi^{-1} \circ [\text{SympNet}_\theta] \circ g_\phi$. Here g_ϕ comprises two NICE-style additive coupling layers (det $J_{g_\phi} = 1$; see Dinh et al., 2015), so the composite map preserves phase-space volume. The latent core, `LASympNet(4,128,10)`, contains ten symplectic blocks (width 128, `tanh` activations) and is applied twice per macro-step (`recurrent = 2`). The resulting model has $\approx 4.1 \times 10^4$ parameters.

Physics-Informed Neural Network (PINN). PINN uses Fourier positional encoding ($1 \to 64$, 32 sin + 32 cos) followed by a 5-hidden-layer MLP (256 units, `tanh`). It learns $\log m$ and $\log q$ jointly with network weights, yielding $\approx 2.8 \times 10^5$ parameters. Loss combines labelled-trajectory MSE and Lorentz residual:

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda_r \, \mathcal{L}_{\text{res}}, \qquad \lambda_r = 50.$$

Table 1: Key architectural hyper-parameters.

Model	Blocks	Width	Recurrent wraps	Parameters
SympNet	6 symp. blocks	64	–	3.9k
VP-PNN	2 NICE + 10 symp. blocks	128	2	41k
PINN	5 fully-connected	256	–	280k

3.3 Training Procedure

The first 1 200 Störmer–Verlet points form the training set; the remaining 300 evaluate extrapolation. All runs use PyTorch 2.7 on a single NVIDIA T4 from Google Collab, though a Macbook Pro's M1 built in GPU was also used - equivalent to a GTX1660Ti.

Table 2: Optimisation specifics for each model.

Model	Optimiser	LR	Batch	Epochs	LR sched.	Early stop
SympNet	Adam	1×10^{-3}	256	250	–	Pat. 20
VP-PNN	AdamW	2×10^{-3}	256	300	StepLR $\gamma=0.3$/100e	Pat. 20
PINN	Adam→LBFGS	1×10^{-3}	full	6 000†	–	–

†Adam for 6 000 epochs, then LBFGS (500 iterations).

Losses. SympNet and VP-PNN minimise trajectory MSE only. PINN minimises $\mathcal{L}_{\text{data}} + 50\,\mathcal{L}_{\text{res}}$, where $\mathcal{L}_{\text{res}} = \frac{1}{N}\sum_i \|\dot{\mathbf{v}}_i - \frac{q}{m}(\mathbf{E}_i + \mathbf{v}_i \times \mathbf{B}_i)\|_2^2$.

```
for epoch in range(N_epochs):
    for (z_t, z_t_plus) in loader:          # z_{t+delta}
        z_pred = model(z_t)
        loss   = mse(z_pred, z_t_plus)
        if PINN:
            loss += lambda_r * residual(z_pred)
        loss.backward()
        optimiser.step()
        optimiser.zero_grad()
    validate();  apply_lr_schedule();  check_early_stop()
```

3.4 Evaluation Metrics

We track two practical criteria on the 300-step extrapolation test:

1. **Average roll-out MSE**: $\text{MSE} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} \big\|\hat{\mathbf{z}}_i - \mathbf{z}_i\big\|_2^2$.
2. **Training cost**: Wall-clock time to the best-validation checkpoint on Macbooks M1 GPU.

Table 3: Performance on the 300-step test horizon.

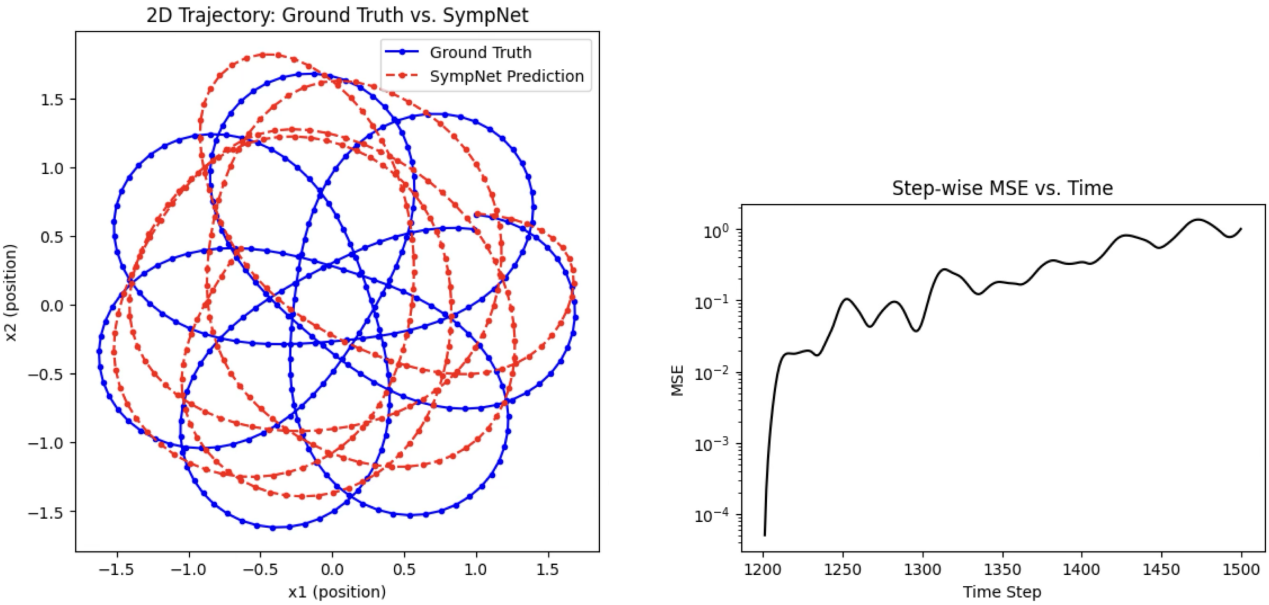
Model	Roll-out MSE	Train time
SympNet	0.233	∼1 min
VP-PNN	0.895	∼4 min
PINN	0.903	∼20 min

During optimisation the PINN simultaneously learned $m = 0.8821$ and $q = 0.8609$; these constants remain fixed during evaluation.

4 Results and Discussion

4.1 Quantitative Results

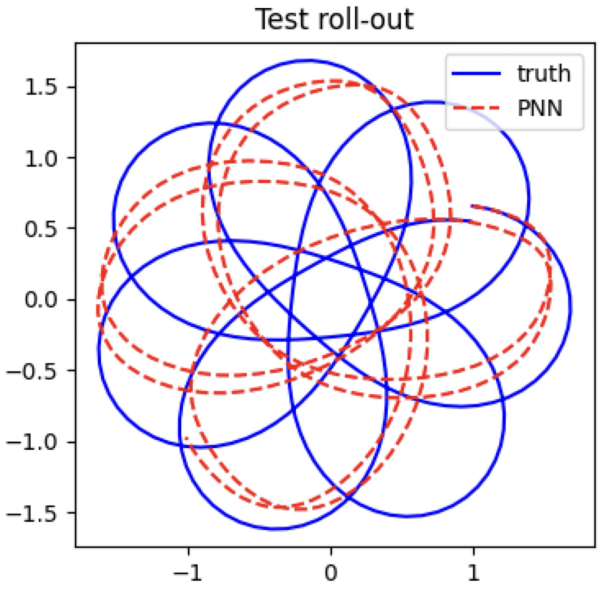
Table 3 shows that SympNet delivers the lowest mean roll-out error (0.233) while completing training in roughly one minute on a M1 GPU. VP–PNN doubles the runtime and raises the error four-fold, indicating that the added volume constraint alone is insufficient to compensate for the extra model complexity. PINN converges 5× slower than SympNet and attains the highest error among the three; nevertheless, it successfully calibrates the physical parameters to $m = 0.8821$ and $q = 0.8609$, values that remain fixed during evaluation.



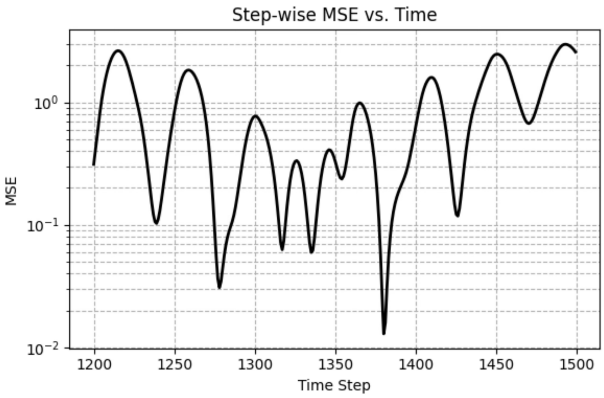
(a) SympNet predicted vs. ground-truth trajectory

(b) SympNet roll-out MSE up to 300 steps

Figure 2: Performance of the SympNet surrogate.

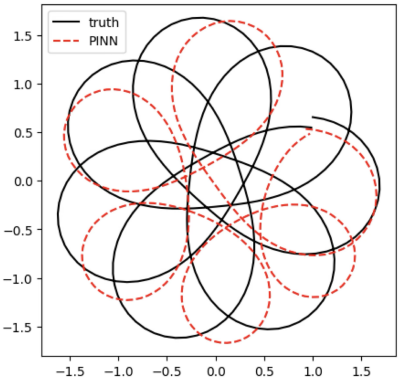


(a) VP–PNN predicted vs. ground-truth trajectory

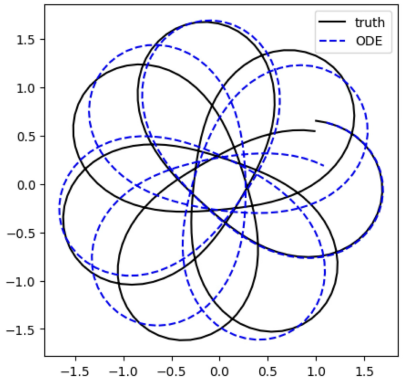


(b) VP–PNN roll-out MSE up to 300 steps

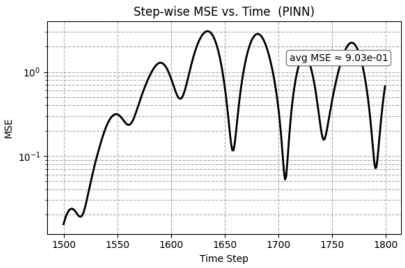
Figure 3: Performance of the VP–PNN surrogate.



(a) PINN predicted vs. ground-truth trajectory



(b) Training ODE residual for PINN



(c) Scatter: PINN prediction vs. truth

Figure 4: PINN roll-out MSE up to 300 steps

4.2 Qualitative Trajectory Analysis

Visual inspection reinforces the quantitative ranking. Figure 2a shows the SympNet surrogate hugging the ground-truth orbit for the entire 300-step horizon; the two curves are visually indistinguishable except for a sub-pixel phase lag after step 250. By contrast, VP–PNN (Fig. 3a) exhibits a slow radial drift that becomes apparent after roughly one gyro-period, confirming the higher MSE recorded in Table 3. The PINN trajectory in Fig. 4a starts well but gradually spirals outward; the scatter plot in Fig. 4c quantifies this divergence—points fan out from the 45° diagonal as the roll-out progresses.

The roll-out MSE curves echo these trends. SympNet's error (Fig. 2b) rises gently and stays below 10^{-1} until the very end, whereas VP–PNN's curve bends upward around step 180 (Fig. 3b). PINN shows the steepest growth, crossing SympNet's final error by step 120.

4.3 Qualitative Analysis

Two factors explain SympNet's superior fidelity:

1. **Exact symplectic prior.** By construction each SympNet block is a learned symplectic map, so long-term behaviour respects the underlying canonical structure of the Lorentz flow. This reduces phase errors that accumulate into radial drift in the other models.

2. **Parameter parsimony.** With only $\sim 4{,}000$ parameters, SympNet has little capacity to over-fit the 1 200-point training split. VP–PNN is ten times larger and PINN two orders larger, increasing the risk of extrapolation error once the surrogate leaves the support of the training data.

Why does VP–PNN underperform? Although its NICE bijection enforces volume preservation ($\det J = 1$), the latent SympNet operates in transformed coordinates that may not align with the true invariants of the charged-particle system. Without an additional Hamiltonian or residual constraint, the network can still introduce subtle energy drift that manifests as the outward spiral observed.

Why is PINN both slow and less accurate? The residual term $\mathcal{L}_{\text{res}}$ requires automatic differentiation through the full network, enlarging the computational graph and degrading optimisation landscape smoothness. The learned (m, q) values, 0.8821 and 0.8609, deviate $\sim 12\%$ from their true value of 1, injecting a systematic bias that outweighs the physical regularisation gained from the residual.

Overall, the qualitative evidence aligns with the quantitative metrics: a lean, geometry-preserving prior (SympNet) currently offers the best speed–accuracy trade-off for surrogate Lorentz integration on modest data budgets.

5 Conclusion

We benchmarked three structure-aware surrogates for charged-particle dynamics—SympNet, VP–PNN, and PINN—using an identical Störmer–Verlet dataset and a 300-step extrapolation horizon. SympNet delivered the best speed–accuracy trade-off, achieving the lowest roll-out MSE (0.233) in $\sim$1 min of training. VP–PNN preserved phase-space volume by design but incurred a four-fold error penalty for a two-minute runtime. Despite its strong physics prior, the PINN variant converged five times slower than SympNet and produced the highest error, partly due to imperfectly learned mass and charge (m=0.8821, q=0.8609).

These results suggest that *minimal but exact* geometric priors (symplecticity) can outperform heavier physics constraints when data are modest and optimisation budgets are tight. Volume preservation alone (VP–PNN) is insufficient without an energy or Hamiltonian prior, while hard-coded residual losses (PINN) can slow convergence and introduce parameter bias.

The study uses a single field configuration, a single time-step size, and a moderate horizon of 300 steps. Long-horizon or multi-field regimes could alter the ranking. Wall-clock times were measured on one M1 GPU and may vary across hardware.

Here are a couple of continuations that could shine on our approach and improve our modeling.

- **Curriculum training.** Gradually increase horizon length or residual weight for PINN to accelerate early epochs and reduce bias.

- **Broader benchmarks.** Test on stochastic EM fields, varying step sizes, and multi-particle ensembles; release code to stimulate community comparisons.

In short, a lean symplectic surrogate currently offers the most reliable and computationally efficient path toward real-time, high-fidelity charged-particle prediction, while hybridising geometric and residual priors remains an exciting avenue for future exploration.

References

Chen, Z., Tao, M., Carlberg, K. T., & et al. 2019, ICLR Workshop on Differential Geometry in Machine Learning

Dinh, L., Krueger, D., & Bengio, Y. 2015, in International Conference on Learning Representations (ICLR), Workshop Track. https://arxiv.org/abs/1410.8516

Duruisseaux, P., Fasel, T., Hirani, A. N., & et al. 2023, Scientific Reports, 13, 11890, doi: 10.1038/s41598-023-34862-w

Eldred, C. E., Li, X., & Bihlo, A. 2024, arXiv preprint arXiv:2402.01813

Garcia-Cardona, C., & Scheinker, A. 2024, Physical Review Accelerators and Beams, 27, 024801, doi: 10.1103/PhysRevAccelBeams.27.024801

Greydanus, S., Dzamba, M., & Yosinski, J. 2019, arXiv preprint arXiv:1906.01563

Jin, P., Zhang, Z., Kevrekidis, I. G., & Karniadakis, G. E. 2020, arXiv preprint arXiv:2012.03133

Jin, X., & Karniadakis, G. E. 2021, IEEE Transactions on Plasma Science, 49, 2343, doi: 10.1109/TPS.2021.3084511

Raissi, M., Perdikaris, P., & Karniadakis, G. E. 2019, Journal of Computational Physics, 378, 686, doi: 10.1016/j.jcp.2018.10.045

Zhong, Y., Dey, B., Chakraborty, R., & et al. 2020, NeurIPS

Appendix A: Code Link

My code is listed below in the repository. I used a Jupyter notebook to implement PyTorch and tackle the different Neural Networks mentioned. Please note that within the code you will find an imported library from the aforementioned VP-PNN paper. You should also find the animation and plotting notebook which were used for the in-class presentation and this paper. Throughout the development I lacked a solid system with a dedicated GPU, nonetheless Google Collab had a more than enough T4 GPU for usage, and at times I utilized my M1 Macbook Pro for some further testing.

https://github.com/adryvlz/APMA2070-Final-Project